OPTIMEM

APPLICATION NOTE AN-002

STARTUP PROCEDURE FOR THE OPTIMEM 1000

January 1985

DISCLAIMER

The authors have taken due care in the preparaton of this document. While every effort has been made to ensure that the information provided is correct, please notify us in the event of an error or inconsistency. Direct your comments to the Marketing Department:

Optimem
435 OAKMEAD PARKWAY
SUNNYVALE, CA 94086
(408) 737-7373

Optimem makes no representation or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantablility or fitness for any purpose.

Further, Optimem reserves the right to revise this publication and to make changes from time to time in contents hereof without obligation to notify any person of such revisions or changes.

****

Startup Procedure for the Optimem 1000 SCSI Interface

The intent of this application note is to illustrate a startup procedure using the SCSI Interface on the Optimem 1000 Optical Disk Drive. The majority of the procedure is standard ANSC SCSI Revision Level 14B, issued in the Spring of 1984. It is assumed the reader is familiar with the standard SCSI. If not, we strongly reccomend that you obtain a copy of the Rev 14B SCSI, by contacting the ANSC X3T9.2 Chairman, William E. Burr, U.S. Department of Commerce, National Bureau of Standards, Technology A-216, Washington D.C. 20234, 301 -921-3723, or X3T9.2 Vice Chairman, John Lohmeyer, NCR Corporation, 3718 N. Rock Road, Wichita, KS 67226, 316 - 688-8703.

Attached to this application note is the 'C' source code to a working program called startup.c which demonstrates this procedure. A copy of the executable code may be obtained by sending a 5-1/4 inch diskette with your request to the Optimem Marketing Group. The executable code runs on an IBM PC or compatible machine having an Adaptive Data Systems Inc. PC-Masterlink host adapter.

This startup procedure assumes the drive is in the initial powered on state. If the drive is not powered on, the Selection Phase will time out as it would for any SCSI ID selected but not present. At the end of this procedure, the drive will be ready to perform reads and writes with a device driver.

> Note: A Completion Status of Busy is returned after the Selection Phase (ie. no Command Phase is entered) if the controller is doing other tasks and cannot accept a command. It is important to check for the Status Phase in the device driver before attempting to send a command.

The following SCSI Commands are used in this procedure:

- Inquiry
- Read Capacity
- Test Unit Ready
- Request Sense
- Start/Stop Unit
- Mode Select
- Mode Sense
- Seek

Inquiry

The first step in the startup procedure is to send the Inquiry command which returns five bytes of data. The Device type Code returned is 04h for the Optimem 1000 signifying a Write Once Read Multiple Device. If the Device Type Code is 7Fh, then the Logical Unit (LUN) requested in the command is not present. These are the only two valid Device Type Codes returned by the Optimem 1000.

Read Capacity

The Read Capacity command is issued in the next step of the procedure. Included in the eight bytes of data returned are the Logical Block Address, which is 00 0F 42 3Fh, and the Logical Block Size, which is 00

00 04 00h. These are the only valid Optimem 1000 responses to the Read Capacity command. Please note that all device classes return eight bytes of data in response to the Read Capacity command, therefore, an Allocation Length is not specified in the command descriptor block.

Test Unit Ready

The Test Unit Ready command is now issued. The Test Unit Ready command returns a Completion Status of 'Good' if the cartridge is in place, spun up, and the controller is ready to process commands. If a Completion Status of Check Condition is returned, a Request Sense command is issued to retrieve the Extended Sense data and determine the cause of the check condition.

Request Sense

The Optimem 1000 SCSI Controller returns up to ten bytes of extended sense data. Data is always returned in response to a Request Sense command, however, the data is valid only after the controller detects a check condition and updates the Extended Sense data registers. Therefore in order to obtain valid sense data, it is imperative a Request Sense command be issued immediately after a Check Condition has been reported.

The Sense Key in byte two should be examined. If it is 06h, a Unit Attention fault has occurred. The most probable cause is that a cartridge has been inserted since the last time the Optimem 1000 was accessed. Another possible cause in systems that support disconnection is that a Reset occurred on the SCSI Bus.

The first eight Sense bytes follow the prescribed format in the ANSC document. Sense bytes nine and ten pertain only to the Optimem Vendor Unique implementations of the Ready Code, ODI Fault Code, and the Controller Fault Code, which further describe the nature of the fault and current state of the drive.

The Ready Code reports the current state of the cartridge. Using this information, it can be determined whether the cartridge is in place, if a Start/Stop Unit command needs to be issued, or a spin up/down is in progress. Consult the Interface Manual for code details.

The Odi Fault Code is not relevant to this discussion.

The Controller Fault Code corresponds to the Sense Key and clarifies the fault condition. At this point in the procedure it should contain a value of 20h to 23h, ie. Not Ready Faults. If the fault is due to a cartridge not inserted, then a prompt is sent to the system operator to load a cartridge.

Start/Stop Unit

The Start/Stop Unit command is issued to spin up the medium. This command has an option of setting an Immed Bit that allows the Completion Status to be returned immediately after receipt of the command. Since the spin up or down process takes up to fifteen seconds, the host can then utilize this time doing other tasks.

The medium can also be spun up from the front panel switch.

The Startup Program again issues a Test Unit Ready and with the return of a Completion Status of 'Good', issues a Mode Select command.

Note that this program checks for one Target or SCSI ID of one and one LUN or Logical Unit of zero. If multiple Targets or multiple LUNs are used, then this procedure would need to be repeated for each SCSI ID and LUN combination.

Mode Select

The Mode Select command is issued to specify controller parameters. Consult the Interface Manual for details.

Mode Sense

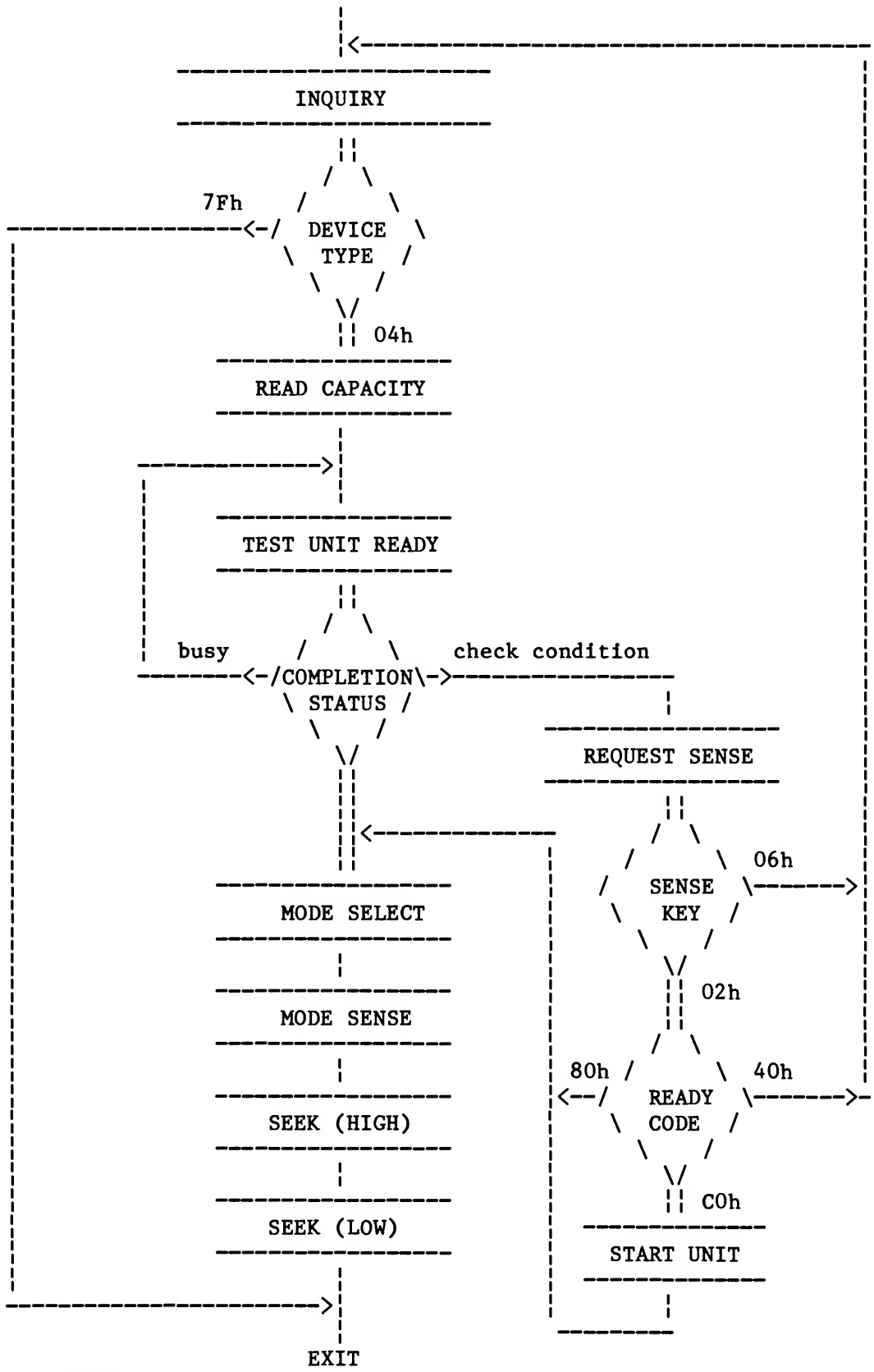
The Mode Sense command is issued to verify the setting of the controller parameters as well as checking the Write Protect (WP) bit which indicates the setting of the write protect tab on the cartridge.

Seek

The last step in the start up procedure performs a seek to a high track and a seek to a low track. These are done to demonstrate that the drive is operational and are not a necessary part of the Startup procedure.

The Optimem 1000 should now be ready for system integration. Note that if the drive has been spun up with the Start/Stop command that it can only be spun down from the host with the Start/Stop command.

FLOW CHART OF STARTUP PROCEDURE



startup.cht 12/13/84

```
/* STARTUP.C
 * Program to bring up the Optimem 1000 Optical Disk Drive per SCSI.
 * OPTIMEM
 * 350 Potrero Avenue
 * Sunnyvale, CA  94086
 * (408) 737-7373
 * L. Lamers
 * version 1.22  12/13/84
 */

#include "stdio.h"
#include "string.h"
#include "SCSIADSI.h"
#include "scsitest.h"

char title[80] = "V:1.22 - STARTUP ";

#define GOOD_STATUS     0x00
#define CHECK_CONDITION 0x02
#define BUSY_CONDITION  0x08
#define WP_BIT          0x80
#define EBC_BIT         0x01

#define MODE_OPTIONS    0x00   /* mode options are Optimem vendor unique */
#define DRT_BIT         0x80   /* and are comprised of the or'd value of */
#define EDL_BIT         0x40   /* this list of bits                      */
#define INTL_BIT        0x20
#define DEDAC_BIT       0x10
#define PEB_BIT         0x08
#define DIAG_BIT        0x04
#define MSS_BIT         0x01

unsigned int wp_bit    = 0;
unsigned int ebc_bit   = 0;    /* 0 to disable blank check on write operations */
                               /* 1 to enable blank check on write operations */

unsigned int drt_bit   = 0;
unsigned int edl_bit   = 0;
unsigned int intl_bit  = 0;
unsigned int dedac_bit = 0;
unsigned int peb_bit   = 0;
unsigned int diag_bit  = 0;
unsigned int mss_bit   = 0;

unsigned int ready_code = 0;  /* Optimem vendor unique status of the cartridge */

unsigned int sense_key = 0;
unsigned int device_type_code = 0;
unsigned int sense_data_length = 0;

main()
{
beginning:     /* label to restart program */

if (inquiry() != SUCCESS)
  phase_error();
```

```
device_type_code = sense_buffer[0] & 0x7f;
if (device_type_code == 0x7f)
  {
  printf ("Logical unit is not present \n");
  exit();
  }
if (device_type_code != 0x04)
  {
  printf ("This is not a Write Once Read Multiple device \n");
  exit();
  }
if (readcapacity() != SUCCESS)
  phase_error();
if (testunitready() != SUCCESS)
  phase_error();
while (cmd->scsi_status == BUSY_CONDITION)
  {
  if (testunitready() != SUCCESS)
    phase_error();
  }
if (cmd->scsi_status == CHECK_CONDITION)
  {
  if (requestsense() != SUCCESS)
    phase_error();
  sense_key = sense_buffer[2] & 0x0f;
  printf (" \n");
  printf ("Sense Key: %02x \n",sense_key);
  printf (" \n");
  if (sense_key == 0x06)
    {
    printf ("Medium changed \n");
    goto beginning;
    }

  if (sense_key == 0x02)
    {
    ready_code = sense_buffer[8] & 0xC0;
    if (ready_code == 0xC0)
      {
      startunit();
      readyloop();
      }
    if (ready_code == 0x80)
      {
      printf ("Spin up or Spin Down in progress \n");
      readyloop();
      }
    if (ready_code == 0x40)
      {
      printf ("Insert cartridge in drive and hit a key & <cr> to continue \n");
      while (getchar() == -1)
        ;

      delay(32000);
      delay(32000);
      goto beginning;
```

```
      }
    } /* end of sense key if */
  } /* end of check condition if */


if (cmd->scsi_status == GOOD_STATUS)
  {
  sense_buffer[2] = ebc_bit;        /* set blank checking option */
  sense_buffer[5] = MODE_OPTIONS;   /* set all other options */
  if (modeselect() != SUCCESS)              /* send the mode option bits */
    phase_error();
  if (cmd->scsi_status == CHECK_CONDITION)
    requestsense();
  if (modesense() != SUCCESS)               /* check mode bits */
    phase_error();

  sense_data_length = sense_buffer[0];      /* get length of sense data */

  ebc_bit = sense_buffer[2] & EBC_BIT;      /* analyze which bits are set */
  if (ebc_bit == EBC_BIT)
    printf ("Blank checking is enabled \n");
  else printf ("Blank checking is disabled \n");

  wp_bit = sense_buffer[2] & WP_BIT;
  if (wp_bit == WP_BIT)
    printf ("Medium is write protected \n");
  else printf ("Medium is write enable \n");

/* the following bits of the MODE SENSE data are vendor unique to Optimem */
  drt_bit = sense_buffer[5] & DRT_BIT;
  if (drt_bit == DRT_BIT)
    printf ("Retries are disabled \n");
  else printf ("Retries are enabled \n");

  edl_bit = sense_buffer[5] & EDL_BIT;
  if (edl_bit == EDL_BIT)
    printf ("Error detection level is disabled \n");
  else printf ("Error detection level is enabled \n");

  intl_bit = sense_buffer[5] & INTL_BIT;
  if (intl_bit == INTL_BIT)
    printf ("Interleave of three is set \n");
  else printf ("Interleave of one is set \n");

  dedac_bit = sense_buffer[5] & DEDAC_BIT;
  if (dedac_bit == DEDAC_BIT)
    printf ("Error detection and correction is disabled - non-EDAC format \n");
  else printf ("Error detection and correction is enabled - EDAC format\n");

  peb_bit = sense_buffer[5] & PEB_BIT;
  if (peb_bit == PEB_BIT)
    printf ("Parity checking enabled \n");
  else printf ("Parity checking disabled \n");

  diag_bit = sense_buffer[5] & DIAG_BIT;
  if (diag_bit == DIAG_BIT)
```

```
    printf ("Diagnostic commands are enabled \n");
  else printf ("Diagnostic commands are disabled \n");

  mss_bit = sense_buffer[5] & MSS_BIT;
  if (mss_bit == MSS_BIT)
    printf ("Side two of the medium is currently accessable \n");
  else printf ("Side one of the medium is currently accessable \n");

  if (seekhigh() != SUCCESS)
    phase_error();
  if (cmd->scsi_status == CHECK_CONDITION)
    requestsense();
  if (seeklow() != SUCCESS)
    phase_error();
  if (cmd->scsi_status == CHECK_CONDITION)
    requestsense();
  } /* end of good status */

} /* end of main() */

readyloop()
  {
  do
    {
    delay (32000);
    testunitready();
    }
  while (cmd->scsi_status == CHECK_CONDITION);
  }

phase_error()
  {
  printf ("Phase error \n");
  reset();
  rezero();
  if (testunitready() != SUCCESS)
    exit();
  }
```